รายการอ้างอิง

ภาษาไทย

ทักษิณ เทพชาตรี. พฤติกรรมและการออกแบบโครงสร้างเหล็ก. กรุงเทพมหานคร: วิศวกรรมสถานแห่งประเทศไทยในพระบรมราชานูปถัมภ์, 2529.

บุญแสง สิริรัตน์ชูวงศ์. การวิเคราะห์โครงข้อแข็งด้วยวิธีอีลาสติก-พลาสติกโดยคำนึงถึงผลจาก P-Δ และการย้อนกลับของโมเมนต์ ณ.จุดหมุนพลาสติก. วิทยานิพนธ์ปริญญามหาบัณฑิต จุฬาลงกรณ์มหาวิทยาลัย, 2535.

ปณิธาน ลักคุณะประสิทธิ์. การวิเคราะห์โครงสร้าง. พิมพ์ครั้งที่ 2, กรุงเทพมหานคร: วิศวกรรมสถานแห่งประเทศไทยในพระบรมราชูปถัมภ์, 2533.

สัญญา เพชรเนียม. การวิเคราะห์โครงข้อแข็งด้วยวิธีอิลาสติก-พลาสติก โดยคำนึงถึงผลจาก P-Δ. วิทยานิพนธ์ปริญญามหาบัณฑิต จุฬาลงกรณ์มหาวิทยาลัย, 2535.

ภาษาอังกฤษ

Bathe, K.J.,and Wilson, E.L. Numerical Methods in Finite Element Analysis. New Jersey: Prentice-Hall, 1976.

Chandra, R.,Krisna, P.,and Trikha, D.N. Elastic-Plastic Analysis of Steel Space Structures. Journal of Structural Division, ASCE, 116(April 1990): 939- 955.

Chen, W.F.,and Lui, E.M. Stability Design of Steel Frames. Florida: CRC press, 1991.

_____. Structural Stability Theory and Implementation. New York: Elsevier, 1978.

Chitti Vijakkhana. Inelastic Stability of Steel Building Frames. Doctoral Dissertation, Asian Institute of technology, 1973.

Goto, Y., and Chen, W.F. Second Order Elastic Analysis for Frame Design. Journal of Structural Division ASCE, 113(July 1987): 1501-1519.

Harrison, H.B. Computer Methods in Structural Analysis. New Jersey: Prentice-Hall, 1965.

Horne, M.R. Plastic Theory of Structures. 2nd ed. Oxford: Pergamon Press,1979.

Jenning, A. ,and Majid, K.I. An Elastic-Plastic analysis by computer for Framed Structures loaded up to collapse. The Structural Engineer 43(December 1965): 407-412.

Kassimali, A. Large Deformation Analysis of Elastic-Plastic Frames. Journal of Structural Division ASCE, 109(August 1983): 1869-1886.

Korn, A.,and Galambos, T.V. Behavior of Elastic-Plastic Frames. Journal of Structural Division ASCE, 94(May 1968): 1119-1142.

Majid, K.I., and Anderson, D. Elastic-Plastic Design of Sway Frames by Computer. Proc. Institute of Civil Engineers 41(December 1968): 705-729.

Neal, B.G. The Plastic Methods of Structural Analysis. 3rd ed. London: Chapman and Hall, 1977.

Salmon, C.G.,and Johnson, J.E. Steel Structure Design and Behavior. 3rd ed. New York: Harper Collins, 1990.

Yang, Y.B. Linear and Nonlinear Analysis of Space Frames with Nonuniform Torsion using Interactive Computer Graphics. Doctoral Dissertation, Cornell University. 1984.

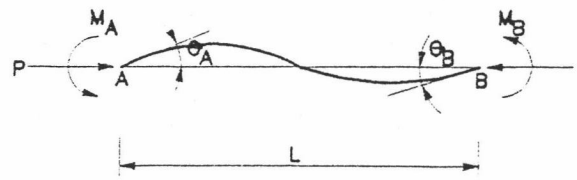
Zienkiewics, O.C., and Taylor, R.L. The Finite Element Method. 4th ed. Vol.2. London: McGrawhill, 1991.

ภาคผนวก

ภาคผนวก ก

การหาฟังก์ชันเสถียรภาพ

การหาฟังก์ชันเสถียรภาพขององค์อาคารคาน-เสาสามารถกระทำได้โดยพิจารณาองค์อาคารมีพื้นที่หน้าตัดเท่ากับ A มีโมเมนต์ของความเฉื่อยของหน้าตัดเท่ากับ I และมีความยาวเท่ากับ L วางอยู่บนฐานรองรับด้านหนึ่งเป็นจุดหมุน (Hinge) ส่วนอีกด้านหนึ่งเป็นแบบจุดหมุนเคลื่อนที่ (Roller) มีแรงกระทำในแนวแกนที่เป็นแรงอัดขนาด P และโมเมนต์กระทำที่ปลายทั้งสองด้านในทิศทางทวนเข็มนาฬิกาขนาด M_A และ M_B ตามลำดับ ดังรูปที่ ก.1



รูปที่ ก.1 องค์อาคารคาน-เสาภายใต้โมเมนต์กระทำที่ปลาย

พิจารณาส่วนขององค์อาคารยาวห่างจากฐานรองรับเท่ากับ x ความสัมพันธ์ของโมเมนต์ที่หน้าตัดกับแรงอื่น ๆ มีค่าเป็นไปตามสมการ (ก.1)

$$M = -M_A - Py + \frac{(M_A + M_B)}{L} \qquad (ก.1)$$

จากความสัมพันธ์ของโมเมนต์กับการเปลี่ยนตำแหน่งตามสมการ (ก.2)

$$M = -EI\, y'' \qquad (ก.2)$$

เมื่อ y'' เป็นอนุพันธ์อันดับที่สองของการเปลี่ยนตำแหน่งเทียบกับระยะตามยาว

E เป็นค่า Modulus of Elasticity

จะได้สมการอนุพันธ์อันดับที่สองของการเปลี่ยนตำแหน่งเทียบกับระยะตามยาวขององค์อาคารดังสมการ (ก.3)

$$EI\,y'' + Py = \frac{(M_A + M_B)}{L}x - M_A \qquad (ก.3)$$

ถ้ากำหนดให้ $k^2 = \frac{P}{EI}$ ความสัมพันธ์ตามสมการ (ก.3) เขียนใหม่ได้เป็น

$$y'' + k^2 y = \frac{(M_A + M_B)}{LEI}x - \frac{M_A}{EI} \qquad (ก.4)$$

คำตอบของสมการคือ

$$y = C_1 \sin kx + C_2 \cos kx + \frac{(M_A + M_B)}{LEIk^2}x - \frac{M_A}{EIk^2} \qquad (ก.5)$$

เมื่อ C_1 และ C_2 เป็นค่าคงที่

พิจารณาเงื่อนไขขอบเขตตามรูปที่ (ก.1)

$$y(0) = 0 \quad , \quad y(L) = 0 \qquad (ก.6)$$

แทนค่าสมการ (ก.6) ในสมการ (ก.5) จะได้

$$C_2 = \frac{M_A}{EIk^2} \qquad (ก.7)$$

$$C_1 = \frac{-1}{EIk^2 \sin kL}(M_A \cos kL + M_B) \qquad (ก.8)$$

แทนค่าตามสมการ (ก.7) และ(ก.8) ลงในสมการ (ก.5) จะได้

$$y = \frac{-(M_A \cos kL + M_B)}{EIk^2 \sin kL}\sin kx + \frac{M_A}{EIk^2}\cos kx + \frac{(M_A + M_B)}{LEIk^2}x - \frac{M_A}{EIk^2} \qquad (ก.9)$$

หรือจัดรูปใหม่ได้เป็น

$$y = \frac{-1}{EIk^2}\left[\frac{\cos kL}{\sin kL}\sin kx - \cos kx - \frac{x}{L} + 1\right]M_A - \frac{1}{EIk^2}\left[\frac{1}{\sin kL}\sin kx - \frac{x}{L}\right]M_B \quad (ก.10)$$

ค่าอนุพันธ์อันดับที่หนึ่งของการเปลี่ยนตำแหน่ง (y') จะมีค่าเป็น

$$y' = -\frac{1}{EIk}\left[\frac{\cos kL}{\sin kL}\cos kx + \sin kx - \frac{1}{kL}\right]M_A - \frac{1}{EIk}\left[\frac{\cos kx}{\sin kL} - \frac{1}{kL}\right]M_B \quad (ก.11)$$

ให้ θ_A และ θ_B เป็นมุมหมุนที่ปลาย A และ B ทิศทางทวนเข็มนาฬิกาซึ่งจะมีค่าเท่ากับอนุพันธ์อันดับที่หนึ่งของการเปลี่ยนตำแหน่งที่จุดนั้น ดังนั้น

$$\theta_A = y'(0) = -\frac{1}{EIk}\left[\frac{\cos kL}{\sin kL} - \frac{1}{kL}\right]M_A - \frac{1}{EIk}\left[\frac{1}{\sin kL} - \frac{1}{kL}\right]M_B$$

$$= \frac{L}{EI}\left[\frac{\sin kL - kL\cos kL}{(kL)^2 \sin kL}\right]M_A + \frac{L}{EI}\left[\frac{\sin kL - kL}{(kL)^2 \sin kL}\right]M_B \quad (ก.12)$$

$$\theta_B = y'(L) = -\frac{1}{EIk}\left[\frac{1}{\sin kL} - \frac{1}{kL}\right]M_A - \frac{1}{EIk}\left[\frac{\cos kL}{\sin kL} - \frac{1}{kL}\right]M_B$$

$$= \frac{L}{EI}\left[\frac{\sin kL - kL}{(kL)^2 \sin kL}\right]M_A + \frac{L}{EI}\left[\frac{\sin kL - kL\cos kL}{(kL)^2 \sin kL}\right]M_B \quad (ก.13)$$

จากสมการ (ก.12) และ (ก.13) เขียนในรูปของเมตริกซ์ได้เป็น

$$\begin{Bmatrix}\theta_A \\ \theta_B\end{Bmatrix} = \begin{bmatrix}f_{11} & f_{12} \\ f_{21} & f_{22}\end{bmatrix}\begin{Bmatrix}M_A \\ M_B\end{Bmatrix} \quad (ก.14)$$

เมื่อ $$f_{11} = f_{22} = \frac{L}{EI}\left[\frac{\sin kL - kL\cos kL}{(kL)^2 \sin kL}\right]$$ (ก.15)

$$f_{21} = f_{12} = \frac{L}{EI}\left[\frac{\sin kL - kL}{(kL)^2 \sin kL}\right]$$ (ก.16)

จากสมการ (ก.14) หาค่าอินเวอร์สของเมตริกซ์จะได้

$$\begin{Bmatrix} M_A \\ M_B \end{Bmatrix} = \begin{bmatrix} f_{11} & f_{12} \\ f_{21} & f_{22} \end{bmatrix}^{-1} \begin{Bmatrix} \theta_A \\ \theta_B \end{Bmatrix}$$ (ก.17)

$$\begin{Bmatrix} M_A \\ M_B \end{Bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \begin{Bmatrix} \theta_A \\ \theta_B \end{Bmatrix}$$ (ก.18)

เมื่อ $$c_{11} = c_{22} = \frac{EI}{L}\left[\frac{kL\sin kL - (kL)^2\cos kL}{2 - 2\cos kL - kL\sin kL}\right]$$ (ก..19)

$$c_{12} = c_{21} = \frac{EI}{L}\left[\frac{(kL)^2 - kL\sin kL}{2 - 2\cos kL - kL\sin kL}\right]$$ (ก..20)

เมื่อรวมการเปลี่ยนตำแหน่งในแนวแกน สมการ (ก.18)เขียนใหม่ได้เป็น

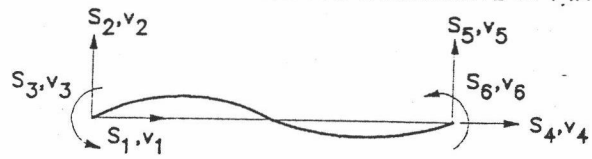
$$\begin{Bmatrix} P \\ M_A \\ M_B \end{Bmatrix} = \frac{EI}{L}\begin{bmatrix} \frac{A}{I} & 0 & 0 \\ 0 & c_{ii} & c_{ij} \\ 0 & c_{ji} & c_{jj} \end{bmatrix} \begin{Bmatrix} e \\ \theta_A \\ \theta_B \end{Bmatrix}$$ (ก.21)

เมื่อ e = การเปลี่ยนตำแหน่งในแนวแกน

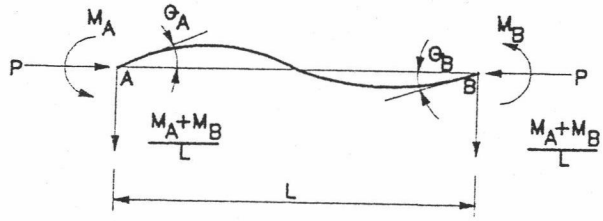
$$c_{ii} = c_{jj} = \left[\frac{kL\sin kL - (kL)^2\cos kL}{2 - 2\cos kL - kL\sin kL}\right]$$ (ก.22)

$$c_{ij} = c_{ji} = \left[\frac{(kL)^2 - kL\sin kL}{2 - 2\cos kL - kL\sin kL}\right]$$ (ก..23)

ก. แรงและการเปลี่ยนตำแหน่งในโคออดิเนตประจำตัว



ข. แรงและการเปลี่ยนตำแหน่งที่อยู่ในรูปแรงอิสระ 3 แรง



รูปที่ ก.2 ความสัมพันธ์ของแรงและการเปลี่ยนตำแหน่งขององค์อาคารคาน - เสา

พิจารณาความสมดุลย์และความสัมพันธ์ของแรงในรูป ก.2 (ก) และ ก.2 (ข)

$$s_1 = P \qquad (ก.24)$$

$$s_2 = \frac{(M_A + M_B)}{L} \qquad (ก.25)$$

$$s_3 = M_A \qquad (ก.26)$$

$$s_4 = -P \qquad (ก.27)$$

$$s_5 = -\frac{(M_A + M_B)}{L} \qquad (ก.29)$$

$$s_6 = M_B \qquad (ก.30)$$

หรือสามารถเขียนในรูปเมตริกซ์ได้เป็น

$$\begin{Bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1/L & -1/L \\ 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 1/L & 1/L \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} P \\ M_A \\ M_B \end{Bmatrix} \qquad (ก.31)$$

ความสัมพันธ์ของการเปลี่ยนตำแหน่งตามรูปที่ ก.2 (ก) และรูปที่ ก.2 (ข) เป็นดังนี้

$$e = -(v_4 + v_5) \qquad (ก.32)$$

$$\theta_A = v_3 + \frac{(v_5 - v_2)}{L} \qquad (ก.33)$$

$$\theta_B = v_6 + \frac{(v_5 - v_2)}{L} \qquad (ก.34)$$

หรือเขียนในรูปเมตริกซ์ได้เป็น

$$\begin{Bmatrix} e \\ \theta_A \\ \theta_B \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1/L & 1 & 0 & 1/L & 0 \\ 0 & -1/L & 0 & 0 & 1/L & 1 \end{bmatrix} \begin{Bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{Bmatrix} \qquad (ก.35)$$

แทนค่าตามสมการ (ก.31) และ (ก.35) ลงในสมการ (ก.21) จะได้

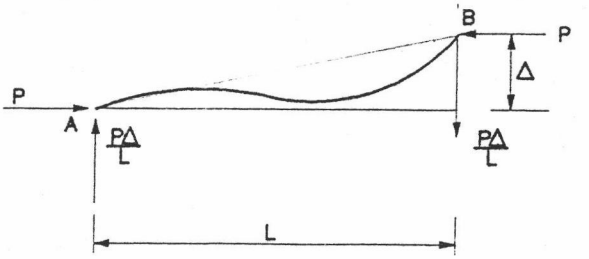
$$\begin{Bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \end{Bmatrix} = \frac{EI}{L} \begin{bmatrix} \frac{A}{I} & 0 & 0 & -\frac{A}{I} & 0 & 0 \\ & \frac{2(c_{ii}+c_{ij})}{L^2} & -\frac{(c_{ii}+c_{ij})}{L} & 0 & -\frac{2(c_{ii}+c_{ij})}{L^2} & -\frac{(c_{ii}+c_{ij})}{L} \\ & & c_{ii} & 0 & \frac{(c_{ii}+c_{ij})}{L} & c_{ij} \\ & & & \frac{A}{I} & 0 & 0 \\ & & \text{sym.} & & \frac{2(c_{ii}+c_{ij})}{L^2} & \frac{(c_{ii}+c_{ij})}{L} \\ & & & & & c_{jj} \end{bmatrix} \begin{Bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{Bmatrix} \qquad (ก.36)$$

หรือเขียนในรูปเมตริกซ์ได้เป็น

$$\{s\}_{ns} = [k]_{ns}\{v\} \qquad (ก.37)$$

เมื่อ $[k]_{ns}$ หมายถึงเมตริกซ์สตีฟเนสที่ไม่พิจารณาผลของการเปลี่ยนตำแหน่งแบบเลื่อนด้านข้าง (Non Sidesway)

เนื่องจากองค์อาคารที่พิจารณาผ่านมาไม่ได้คำนึงถึงการเปลี่ยนตำแหน่งในทิศทางตั้งฉากกับองค์อาคารดังนั้นเพื่อให้ได้สตีฟเนสที่สมบูรณ์จะพิจารณาผลของการเปลี่ยนตำแหน่งตั้งฉากกับองค์อาคารได้โดยพิจารณาองค์อาคารตามรูปที่ ก.1 แต่มีค่าการเปลี่ยนตำแหน่งที่ในทิศทางตั้งฉากกับองค์อาคารที่ปลาย B ขนาดเท่ากับ Δ ดังแสดงในรูปที่ ก.3



รูปที่ ก.3 ผลของการเปลี่ยนตำแหน่งด้านข้างต่อการเพิ่มของแรงเฉือน

จากรูปที่ ก.3 จะเห็นว่า

$$\Delta = (d_2 - d_5) \qquad (ก.38)$$

$$\begin{Bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \end{Bmatrix} = \frac{EI}{L} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ & -\frac{P}{L} & 0 & 0 & \frac{P}{L} & 0 \\ & & 0 & 0 & 0 & 0 \\ & & & 0 & 0 & 0 \\ & \text{sym.} & & & -\frac{P}{L} & 0 \\ & & & & & 0 \end{bmatrix} \begin{Bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{Bmatrix} \qquad (ก.39)$$

หรือ

$$\{s\}_s = [k]_s \{v\} \qquad (ก.40)$$

เมื่อ $[k]_s$ หมายถึงเมตริกซ์สตีฟเนสที่พิจารณาผลของการเปลี่ยนตำแหน่งแบบ เลื่อนด้านข้าง (Sidesway)

ดังนั้นความสัมพันธ์ของแรงกับการเปลี่ยนตำแหน่งที่เกิดขึ้นทั้งหมดจะได้

$$\{s\} = [[k]_{ns} + [k]_s]\{v\} \qquad (ก.41)$$

$$\{s\} = [k(v)]\{v\} \qquad (ก.42)$$

เมื่อ $$[k(v)] = [[k]_{ns} + [k]_s] \qquad (ก.43)$$

หรือเขียนใหม่ได้เป็น

$$[k(v)] = \frac{EI}{L}\begin{bmatrix} \frac{A}{I} & 0 & 0 & -\frac{A}{I} & 0 & 0 \\ & \frac{12\Phi_1}{L^2} & -\frac{6\Phi_2}{L} & 0 & -\frac{12\Phi_1}{L^2} & -\frac{6\Phi_2}{L} \\ & & 4\Phi_3 & 0 & \frac{6\Phi_2}{L} & 2\Phi_4 \\ & & & \frac{A}{I} & 0 & 0 \\ & \text{sym.} & & & \frac{12\Phi_1}{L^2} & \frac{6\Phi_2}{L} \\ & & & & & 4\Phi_3 \end{bmatrix}$$

(ก.45)

เมื่อ

$$\Phi_1 = \frac{(kL)^3 \sin kL}{12(2-2\cos kL - kL\sin kL)} \qquad (ก.46)$$

$$\Phi_2 = \frac{(kL)^2(1-\cos kL)}{6(2-2\cos kL - kL\sin kL)} \qquad (ก.47)$$

$$\Phi_3 = \frac{(kL)(\sin kL - kL\cos kL)}{4(2-2\cos kL - kL\sin kL)} \qquad (ก.48)$$

$$\Phi_4 = \frac{(kL)(kL - \sin kL)}{2(2-2\cos kL - kL\sin kL)} \qquad (ก.49)$$

สมการ (ก.46) ถึง (ก.49) เป็นการพิจารณากรณีที่แรงในแนวแกนเป็นแรงอัด แต่ถ้าหากแรงในแนวแกนเป็นแรงดึงแรงในแนวแกน (P) จะถูกแทนที่โดยแรงที่มีเครื่องหมายเป็นลบ (- P) ดังนั้นในสมการ (ก.46) ถึง (ก.49) ค่าของ kL จะถูกแทนที่โดย i kL เมื่อ $i = \sqrt{-1}$

จากคุณสมบัติทางคณิตศาสตร์

$$-i \sin ikL = \sinh kL \qquad (ก.50)$$

$$\cos ikL = \cosh kL \qquad (ก.51)$$

แทนค่าตามสมการ (ก.50) และ (ก.51) ลงใน (ก.46) ถึง (ก.49) จะได้

$$\Phi_1 = \frac{(kL)^3 \sinh kL}{12(2 - 2\cosh kL + kL \sinh kL)} \qquad (ก.52)$$

$$\Phi_2 = \frac{(kL)^2 (\cosh kL - 1)}{6(2 - 2\cosh kL + kL \sinh kL)} \qquad (ก.53)$$

$$\Phi_3 = \frac{(kL)(kL \cosh kL - \sinh kL)}{4(2 - 2\cosh kL + kL \sinh kL)} \qquad (ก.54)$$

$$\Phi_4 = \frac{(kL)(\sinh kL - kL)}{2(2 - 2\cosh kL + kL \sinh kL)} \qquad (ก.55)$$

สมการ (ก.46) ถึง (ก.49) และ สมการ (ก.52) ถึง (ก.55) เรียกว่า ฟังก์ชันเสถียรภาพ (Stability Functions)

จากฟังก์ชันเสถียรภาพที่ได้จะเห็นว่าในกรณีที่แรงในแนวแกนมีทิศทางต่างกันจำเป็นที่จะต้องใช้ฟังก์ชันต่างกันทำให้ยุ่งยาก เพื่อแก้ปัญหานี้ จะใช้การกระจายฟังก์ชันโดยอาศัยอนุกรมเทเลอร์ ซึ่งจะสามารถใช้อนุกรมเพียงชุดเดียวแทนฟังก์ชันได้ทั้งกรณีแรงในแนวแกนเป็นแรงอัดและแรงดึง ความสัมพันธ์ของอนุกรมเทลอร์แสดงได้ดังสมการ (ก.56)

$$f(z) = f(a) + f'(a)(z-a) + \frac{f''}{2!}(z-a)^2 + \ldots + \frac{f^{n-1}(a)}{(n-1)!}(z-a)^{n-1} + \ldots \qquad (ก.56)$$

เมื่อ $f(z)$ ฟังก์ชันของตัวแปร z ใด ๆ

a จุดที่กำหนดให้ สำหรับงานวิจัยนี้ใช้ค่าเท่ากับ 0

$f^i(a)$ อนุพันธ์อันดับที่ i ที่จุด $z = a$

n จำนวนเทอมของอนุกรมที่ต้องการ

แทนค่าสมการ (ก.46) ถึง (ก.49) และสมการ (ก.51) ถึง (ก.55) ลงในสมการ (ก.56) จะได้

$$\Phi_1 = 1 - \frac{1}{10}N - \frac{1}{8400}N^2 - \frac{1}{756000}N^3 - \frac{37}{2328480000}N^4 - \ldots \qquad (ก.57)$$

$$\Phi_2 = 1 - \frac{1}{60}N - \frac{1}{8400}N^2 - \frac{1}{756000}N^3 - \frac{37}{2328480000}N^4 + \ldots \qquad (ก.58)$$

$$\Phi_3 = 1 - \frac{1}{30}N - \frac{11}{25200}N^2 - \frac{1}{108000}N^3 - \frac{509}{2328480000}N^4 + \ldots \qquad (ก.59)$$

$$\Phi_4 = 1 + \frac{1}{60}N + \frac{13}{25200}N^2 + \frac{11}{756000}N^3 + \frac{907}{2328480000}N^4 + \ldots \qquad (ก.60)$$

เมื่อ

$$N = (kL)^2 = \frac{PL^2}{EI} \qquad (ก.61)$$

P หมายถึง แรงในแนวแกนซึ่งมีค่าเป็นบวกเมื่อเป็นแรงอัด หรือมีค่าเป็นลบเมื่อเป็นแรงดึง

Listing ของโปรแกรมคอมพิวเตอร์

โปรแกรมคอมพิวเตอร์นี้พัฒนาโดยใช้ภาษา C โดยใช้คอมไพเลอร์ Turbo C Version 2.0[1] ในการสร้างแฟ้มโปรแกรมใช้งาน (Execute File) ซึ่งพัฒนาขึ้นโดยอาศัยทฤษฎีและแนวคิดในการวิเคราะห์โครงข้อแข็งที่กล่าวถึงในบทที่ 2 และใช้ขั้นตอนการทำงานของโปรแกรมคอมพิวเตอร์ที่กล่าวถึงในบทที่ 3 การเก็บค่าตัวแปรที่สำคัญจะแสดงรายละเอียดไว้ในประกาศ (Comment) ไว้ในตำแหน่งที่มีการกำหนดค่า (Declare) ของตัวแปรนั้นๆ

รายละเอียดของโปรแกรมมีดังต่อไปนี้

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <conio.h>
#include <graphics.h>
#include <bios.h>
struct co_or{int node_no;
             double x;
             double y;
             int bou[3];
            };
/* node_no   = Node Number
   x         = Location of Node in X Direction Related to Global Coordinate
   y         = Location of Node in Y Direction Related to Global Coordinate
  bou[3]     = Variable Store Boudary Condition Code Define by Program As Per Data Input
*/
```

---

[1]Herbert Schildt. Turbo C/C++: The Complete Reference., 2nd Ed., California: McGraw-Hill, 1992.

```
struct ele_con{int ele_no;
               int node_1;
               int node_2;
               double Es;
               double I;
               double Z;
               double A;
               int hinge_1;
               int hinge_2;
              };
/* ele_no        = Element Number
   node_1        = 1st Node
   node_2        = 2nd Node
   Es            = Modulus of Elasticity
   I             = Moment of Inertia
   Z             = Plastic Modulus
   A             = Sectional Area
   hinge_1       = Hinge Code on 1st Node
   hinge_2       = Hinge Code on 2nd Node
*/
struct load{int node_no;
            double x_load;
            double y_load;
            double moment;
           };
/* node_no       = Node Number which was  Loaded
   x_load        = Load in X Direction
   y_load        = Load in Y Direction
   moment        = Moment
*/
int mainwin(int menu_status);
void inputdat(char *fn);
int sol_menu(void);
void solution(int mode,char *fn);    /*solution mode*/
void result(char *fn);
```

```
void main(void)      /* Main Function */
{
int mode1,sol_mode,menu_status=0;
char fn[15];   /* fn[15]= File Name */
normvideo();
for(;;){
   mode1=mainwin(menu_status);
     switch(mode1){
     case 0:inputdat(fn);        /* Goto Input Data Mode */
              break;
     case 1:sol_mode=sol_menu();
              if(sol_mode!=4)solution(sol_mode,fn);   /* Goto Solution Mode */
              break;
     case 2:result(fn);   /*  Show Result */
              break;
     default:return;
     }
  menu_status=mode1;
}
}
void w_string(char *stp,int x,int y,int backgroundcol);
void border(int startx,int starty,int endx,int endy);
int get_response(int m,int x,int y,int count,char *menu[],char *keys,int step,int dir_check);
int mainwin(int menu_status)    /* Function for Shown Main Menu on Screen */
{
int i,norm_backgroundcol=0,rev_backgroundcol=1;
int step=20,dir_check=1;
char *st[]={
   "Input data",
   "Solution",
   "Result",
   "Exit",
   };
char keys[]="isre";
clrscr();
window(2,4,78,25);
```

```
 border(2,4,78,25);
 gotoxy(1,1);
 putch(255);
 for(i=1;i<75;i++)putch(219);
 putch(255);
 /*initialize text attribute & background color*/
 for(i=0;i<4;i++)w_string(st[i],5+i*step,1,norm_backgroundcol);
 for(i=0;i<4;i++)if(i==menu_status)w_string(st[i],5+i*step,1,rev_backgroundcol);
 i=get_response(menu_status,5,1,4,st,keys,step,dir_check);
 return(i);
 }
 void w_string(char *stp,int x,int y,int backgroundcol)
 {
 char c;
 int i;
 gotoxy(x,y);
 textattr(REDlbackgroundcol*16);
 c=stp[0];
 putch(c);
 for(i=1;i<strlen(stp);i++){
   textattr(WHITElbackgroundcol*16);
   c=stp[i];
   putch(c);
}
 normvideo();
}
 void border(int startx,int starty,int endx,int endy)
 {
 register int i;
 gotoxy(1,1);
 gotoxy(1,2);
 putch(213);
 for(i=1;i<endx-startx-1;i++)putch(205);
 putch(184);
 gotoxy(1,endy-starty);
 putch(192);
```

```
 for(i=1;i<endx-startx-1;i++)putch(196);
 putch(217);
 for(i=3;i<endy-starty;i++){
   gotoxy(1,i);
   putch(179);
   gotoxy(endx-startx,i);
   putch(179);
 }
 gotoxy(3,2);
 return;
 }
 int is_in(char *s,char c);
 int get_response(int menu_status,int x,int y,int count,char *menu[],char *keys,int step,int dir_check)
 {
 union inkey{
     char ch[2];
     int i;
     }c;
int arrow=0,arrowx_step=0,arrowy_step=0,key;
int norm_backgroundcol=0,rev_backgroundcol=1;
arrow=arrow+menu_status;
if(dir_check==1)arrowx_step=arrowx_step+step*menu_status;
if(dir_check==0)arrowy_step=arrowy_step+step*menu_status;
gotoxy(x,y);
for(;;){
  while(!bioskey(1));
  c.i=bioskey(0);
  gotoxy(x+arrowx_step,y+arrowy_step);
  w_string(menu[arrow],x+arrowx_step,y+arrowy_step,norm_backgroundcol);
  if(c.ch[0]){
    key=is_in(keys,tolower(c.ch[0]));
    if(key)return(key-1);
    switch(c.ch[0]){
       case '\r':return(arrow);
       case '27':return (-1);
       }
```

```
 }else
 {
    switch(c.ch[1]){
      case 72:     /*Up Arrow */
            if(dir_check==0){
              arrow--;
              arrowy_step=arrowy_step-step;
              }
            break;
      case 80:     /*Down Arrow */
            if(dir_check==0){
              arrow++;
              arrowy_step=arrowy_step+step;
              }
            break;
      case 75:     /*Left Arrow */
            if(dir_check==1){
              arrow--;
              arrowx_step=arrowx_step-step;
              }
            break;
      case 77:     /*Right Arrow */
            if(dir_check==1){
              arrow++;
              arrowx_step=arrowx_step+step;
              }
            break;
    }
} /*end if*/
if(arrow==count){
    arrow=0;
   if(dir_check==0)arrowy_step=0;
   if(dir_check==1)arrowx_step=0;
}
if(arrow<0){
    arrow=count-1;
```

```
        if(dir_check==0)arrowy_step=step*(count-1);
        if(dir_check==1)arrowx_step=step*(count-1);
    }

    gotoxy(x+arrowx_step,y+arrowy_step);
    w_string(menu[arrow],x+arrowx_step,y+arrowy_step,rev_backgroundcol);
  }
  }
  int is_in(char *s,char c)
  {
  register i;
    for(i=0;*s;i++)if(*s++==c)return(i+1);
    return(0);
  }
  int input_submenu(char *fn,int l);
  void input_data(char *fn,int i,int j);
  void inputdat(char *fn)
  {
  int i,j,k,l,norm_backgroundcol=0,rev_backgroundcol=1;
  int step=2,dir_check=0,menu_status=0;
  char *st[]={
    "Revise Exist File",
    "Edit New File",
    "Quit",
    };
 char keys[]="req";
 l=0;
 for(;;){
   if(l==0||j==4){
     window(5,5,28,14);
     border(5,5,28,14);
     /*initialize text attribute & background color*/
     for(k=0;k<3;k++)w_string(st[k],3,3+k*step,norm_backgroundcol);
     for(k=0;k<3;k++)if(k==menu_status)w_string(st[k],3,3+k*step,rev_backgroundcol);
     i=get_response(menu_status,3,3,3,st,keys,step,dir_check);
  }
```

```
        switch(i){
                case 0:
                    j=input_submenu(fn,l);
                    input_data(fn,i,j);
                    break;
                case 1:
                    j=input_submenu(fn,l);
                    input_data(fn,i,j);
                    break;
                default:
                    return;
        }
    menu_status=i;
    l++;
}
}
int input_submenu(char *fn,int l)
{
int i,j,norm_backgroundcol=0,rev_backgroundcol=1;
int step=2,dir_check=0,menu_status=0;
char *st[]={
    "Input Control Parameter",
    "Input Node Data",
    "Input Element Data",
    "Input Load data",
    "Quit"
    };
char keys[]="req";
window(30,10,60,23);
border(30,10,60,23);
if(l==0){
    gotoxy(3,3);
    cprintf("Input File Name:");
    cscanf("%s",fn);
}
/*initialize text attribute & background color*/
```

```
for(i=0;i<5;i++)w_string(st[i],3,3+i*step,norm_backgroundcol);
for(i=0;i<5;i++)if(i==menu_status)w_string(st[i],3,3+i*step,rev_backgroundcol);
i=get_response(menu_status,3,3,5,st,keys,step,dir_check);
return(i);
}
void input_dir(FILE *fp,int mode2);
void input_coo(FILE *fp,int mode2);
void input_ele(FILE *fp,int mode2);
void input_ld(FILE *fp,int mode2);
void input_data(char *fn,int mode1,int mode2)
{
register int i,j;
FILE *fp;
char fn1[15];
window(3,4,77,22);
border(3,4,77,22);
for(i=2;i<74;i++){
   for(j=3;j<18;j++){
       gotoxy(i,j);
       printf(" ");
  }
}
gotoxy(3,16);
cprintf("Prof.Dr. Taksin Thepchatri ");
gotoxy(3,17);
cprintf("Mr. Winai Gaewgoontol ");
strcpy(fn1,fn);
switch(mode2){
   case 0:
          strcat(fn1,".dir");
          if(mode1==0)fp=fopen(fn1,"r+");
          else if(mode1==1)fp=fopen(fn1,"w");
          input_dir(fp,mode2);
          break;
   case 1:
          strcat(fn1,".co");
```

```
            if(mode1==0)fp=fopen(fn1,"r+");
            else if(mode1==1)fp=fopen(fn1,"w");
            input_coo(fp,mode2);
            break;
    case 2:
            strcat(fn1,".el");
            if(mode1==0)fp=fopen(fn1,"r+");
            else if(mode1==1)fp=fopen(fn1,"w");
            input_ele(fp,mode2);
            break;
    case 3:
            strcat(fn1,".lo");
            if(mode1==0)fp=fopen(fn1,"r+");
            else if(mode1==1)fp=fopen(fn1,"w");
            input_ld(fp,mode2);
            break;
    default:
            return;
}
fclose(fp);
return;
}
void input_dir(FILE *fp,int mode2)
{
register int i,j,k;
char s;
gotoxy(3,3);
cprintf("PROJECT TITLE :");
gotoxy(3,4);
cprintf("ENGINEER :");
gotoxy(3,5);
cprintf("Force Unit :");
gotoxy(3,6);
cprintf("Length Unit :");
gotoxy(3,7);
cprintf("DD/MM/YY :");
```

```
 gotoxy(19,3);
 cscanf("%s",&s);
 fprintf(fp,"%s\\\n",s);
 gotoxy(14,4);
 cscanf("%s",&s);
 fprintf(fp,"%s\\\n",s);
 gotoxy(16,5);
 cscanf("%s",&s);
 fprintf(fp,"%s\\\n",s);
 gotoxy(17,6);
 cscanf("%s",&s);
 fprintf(fp,"%s\\\n",s);
 gotoxy(14,7);
 cscanf("%s",&s);
 fprintf(fp,"%s\\\n",s);
 getch();
 return;
}
void input_coo(FILE *fp,int mode2)
{
}
void input_ele(FILE *fp,int mode2)
{
}
void input_ld(FILE *fp,int mode2)
{
}
int sol_menu(void)
{
register int i,norm_backgroundcol=0,rev_backgroundcol=1;
int step=2,dir_check=0;
char *st[]={
   "1st Order Elastic Analysis",
   "2nd Order Elastic Analysis",
   "1st Order Elastic-Plastic Analysis",
   "2nd Order Elastic-Plastic Analysis",
```

```
    "Quit",
    };
  char keys[]="";
  window(25,5,65,17);
  border(25,5,65,17);
  /*initialize text attribute & background color*/
  w_string(st[0],3,3,rev_backgroundcol);
  w_string(st[1],3,5,norm_backgroundcol);
  w_string(st[2],3,7,norm_backgroundcol);
  w_string(st[3],3,9,norm_backgroundcol);
  w_string(st[4],3,11,norm_backgroundcol);
  i=get_response(0,3,3,5,st,keys,step,dir_check);
  clrscr();
  return(i);
  }
  void on_screen_file(char *fn);
  void print_graph(int i);
  void result(char *fn)
  {
  register int i,norm_backgroundcol=0,rev_backgroundcol=1;
  int step=2,dir_check=0;
  char *st[]={
    "On Screen File",
    "On screen Graphics",
    "On Printer File",
    "On Printer Graphics",
    "Quit",
    };
  char keys[]="";
  window(28,5,52,17);
  border(28,5,52,17);
  /*initialize text attribute & background color*/
  w_string(st[0],3,3,rev_backgroundcol);
  w_string(st[1],3,5,norm_backgroundcol);
  w_string(st[2],3,7,norm_backgroundcol);
  w_string(st[3],3,9,norm_backgroundcol);
```

```
w_string(st[4],3,11,norm_backgroundcol);
i=get_response(0,3,3,5,st,keys,step,dir_check);
switch(i){
   case 0:on_screen_file(fn);
          break;
   case 1:on_screen_file(fn);
          break;
   case 2:print_graph(1);
          break;
   case 3:print_graph(1);
          break;
}
}
void on_screen_file(char *fn)
{
int no_node,no_ele,i,j,k,l,m,n,o;
float a,lf;
char *b,c,f1[15],f2[15];
FILE *fp1,*fp2;
strcpy(f1,fn);
strcpy(f2,fn);
strcat(f1,".str");
strcat(f2,".lfd");
fp1=fopen(f1,"r");
fp2=fopen(f2,"r");
clrscr();
window(3,4,77,22);
border(3,4,77,22);
gotoxy(3,3);
cprintf("Select File:\n\r");
gotoxy(3,4);
cprintf("(1) Internal Member Force at Plastic Hinge Form\n\r");
gotoxy(3,5);
cprintf("(2) Displacement");
scanf("%i",&i);
switch(i){
```

```
case 0:
case 1:
  fscanf(fp1,"%i\\",&no_ele);
  do{
        fscanf(fp1,"%f\\",&lf);
         if(lf>0){
            fscanf(fp1,"%i\\",&k);
            gotoxy(3,3);
            cprintf("Project Name : %s\n\r",fn);
            gotoxy(3,4);
            cprintf("Load Factor=%9.3f  :Hinge No=%i                    \n\r",lf,k);
            j=0;
            o=5;
            gotoxy(3,5);
            cprintf("                                                   ");
            gotoxy(3,6);
            cprintf("                                                ");
            do{
              if(j>=3){
                   o++;
                   j=0;
              }
              gotoxy(3+20*j,o);
              fscanf(fp1,"%i\\",&k);
              fscanf(fp1,"%c\\",&c);
              if(k>0){
                     if(c=='N')cprintf("Ele No=%i:End=Near\\",k);
                     else cprintf("Ele No=%i:End=Far\\",k);
              }
              ++j;
            }while(k>0);
            gotoxy(3,7);
            cprintf("EleNo:  Axial  :  Shear  :  Bending :  Axial  :  Shear  :  Bending\n\r");
            j=8;
            n=1;
            do{
```

```
                gotoxy(3,j);
                cprintf("%5i",n);
                m=0;
                for(l=0;l<6;l++){
                    fscanf(fp1,"%f\\",&a);
                    gotoxy(9+11*m,j);
                    cprintf("%10.4e",a);
                    m++;
                }
              j++;
              if((n%10)==0&&(n!=0)){
                  j=8;
                  getch();
                  getch();
                  for(l=j;l<j+10;l++){
                    gotoxy(3,l);
                    cprintf("                                                    \n\r");
                  }
                  j=8;
              }
              n++;
              }while(n<=no_ele);
  getch();
   }
  }while(lf>0);
  break;
case 2:
  fscanf(fp2,"%i\\",&no_node);
  do{
      fscanf(fp2,"%f\\",&lf);
       if(lf>=0){
          gotoxy(3,3);
          cprintf("Project Name : %s\n\r",fn);
          gotoxy(3,4);
          cprintf("Load Factor=%9.3f                    \n\r",lf);
          gotoxy(3,5);
```

```
                cprintf("Node No:    X-Disp    :    Y-Disp    :    Rotation    \n\r");
                k=0;
                for(i=0;i<no_node;i++){
                    gotoxy(3,6+k);
                    cprintf("%7i",1+i);
                    for(j=0;j<3;j++){
                       fscanf(fp2,"%f\\",&a);
                       gotoxy(11+16*j,6+k);
                       cprintf("%15.7e",a);
                    }
                    k++;
                    if((i%10)==0&&i!=0){
                        k=0;
                        getch();
                              for(j=0;j<=10;j++){
                                 gotoxy(3,6+j);
                                 cprintf("                                                    ");
                              }
                    }
                }
                getch();
            }
        }while(lf>=0);
        break;
}
fclose(fp1);
fclose(fp2);
}
void print_graph(int i)
{
}
int get_neq(FILE *fp);
struct co_or *alloc_co(int n);
void get_node(FILE *fp,int n,struct co_or *pt);
struct ele_con *alloc_ele(int n);
void get_ele(FILE *fp,int n,struct ele_con *pt);
```

```
struct load *alloc_ld(int n);
void get_ld(FILE *fp,int n,struct load *pt);
void solving(char *fname,int mode,double fy,int no_node,int no_ele,int no_ld,struct co_or *pt_1,struct ele_con
*pt_2,struct load *pt_3);
void solution(int mode,char *fname)   /* Function for Getting Data to the Memory */
{
 struct co_or *pt_co;  /*   *pt_co    = pointer for Point Coordinate Data In  Memory */
 struct ele_con *pt_ele;     /*        *pt_ele   = Pointer for Point Element Data In Memory */
 double fy;          /* fy     = Yield Strength */
 struct load *pt_ld;         /*        *pt_ld    = Pointer for Pointer Load Data In Memory */
 int i,n_node,n_ele,n_ld,j;    /*    n_node = Number of  Node
                                    n_ele   = Number of Element
                                    n_ld    =  Number of Load
                               */
FILE *fp;
char fname0[15],fname1[15],fname2[15];
  strcpy(fname0,fname);
  strcpy(fname1,fname);
  strcpy(fname2,fname);
  /* input node data */
  strcat(fname0,".coo");
  fp=fopen(fname0,"r");
    n_node=get_neq(fp);
    pt_co=alloc_co(n_node);
    get_node(fp,n_node,pt_co);
 fclose(fp);
 /* input element data */
 strcat(fname1,".ele");
 fp=fopen(fname1,"r");
 fscanf(fp,"%lf\",&fy);
   n_ele=get_neq(fp);
   pt_ele=alloc_ele(n_ele);
   get_ele(fp,n_ele,pt_ele);
 fclose(fp);
 /* input load data */
 strcat(fname2,".lds");
```

```
    fp=fopen(fname2,"r");
      n_ld=get_neq(fp);
      pt_ld=alloc_ld(n_ld);
      get_ld(fp,n_ld,pt_ld);
    fclose(fp);
    solving(fname,mode,fy,n_node,n_ele,n_ld,pt_co,pt_ele,pt_ld);
    free(pt_co);
    free(pt_ele);
    free(pt_ld);
}
int get_neq(FILE *fp)
{
  int neq;
  fscanf(fp,"%i\",&neq);
  return(neq);
}
struct co_or *alloc_co(int n)
{
struct co_or *pt;
pt=(struct co_or *)malloc(n*sizeof(struct co_or));
if(pt==NULL){
    printf("Out of Memory.\n");
    exit(1);
}
return(pt);
}
void get_node(FILE *fp,int n,struct co_or *pt)
{
register int i,j;
  for(i=0;i<n;i++){
      fscanf(fp,"%i\",&pt[i].node_no);
      fscanf(fp,"%lf\",&pt[i].x);
      fscanf(fp,"%lf\",&pt[i].y);
      for(j=0;j<3;j++){
            fscanf(fp,"%i\",&pt[i].bou[j]);
      }
```

```
  }
}
struct ele_con *alloc_ele(int n)
{
struct ele_con *pt;
pt=(struct ele_con *)malloc(n*sizeof(struct ele_con));
if(pt==NULL){
   printf("Out of Memory.\n");
   exit(1);
}
return(pt);
}
void get_ele(FILE *fp,int n,struct ele_con *pt)
{
register int i;
  for(i=0;i<n;i++){
      fscanf(fp,"%i\",&pt[i].ele_no);
      fscanf(fp,"%i\",&pt[i].node_1);
      fscanf(fp,"%i\",&pt[i].node_2);
      fscanf(fp,"%lf\",&pt[i].Es);
      fscanf(fp,"%lf\",&pt[i].I);
      fscanf(fp,"%lf\",&pt[i].Z);
      fscanf(fp,"%lf\",&pt[i].A);
      fscanf(fp,"%i\",&pt[i].hinge_1);
      fscanf(fp,"%i\",&pt[i].hinge_2);
  }
}
struct load *alloc_ld(int n)
{
struct load *pt;
pt=(struct load *)malloc(n*sizeof(struct load));
if(pt==NULL){
   printf("Out of Memory.\n");
   exit(1);
}
return(pt);
```

```
}
void get_ld(FILE *fp,int n,struct load *pt)
{
register int i;
  for(i=0;i<n;i++){
      fscanf(fp,"%i\",&pt[i].node_no);
      fscanf(fp,"%lf\",&pt[i].x_load);
      fscanf(fp,"%lf\",&pt[i].y_load);
      fscanf(fp,"%lf\",&pt[i].moment);
  }
return;
}
struct dof{int x_dof;
            int y_dof;
            int theta_dof;
           };
struct m_stress{double lf;
                  double s[6];
                };
int get_dof(int no_node,int no_ele,struct co_or *pt_co,struct ele_con *pt_ele);
void gen_dof_no(int no_node,int no_dof,struct co_or *pt_co,struct dof *pt);
void print_disp(double lf,FILE *fp,int no_node,struct dof *dof_pt,double *pt_v);
int *alloc_h(int n);
void get_h(int no_dof,int *pt_h,int no_ele,struct ele_con *pt_ele,struct dof *dof_pt);
double *alloc_K(int h);
void get_k(int no_ele,int *pt_h,double *pt_k,struct co_or *pt_co,struct ele_con *pt_ele,struct dof *dof,struct
m_stress *pt,int order_index);
void chk_k(int no_dof,int *pt_h,double *pt_k);
void get_r(int n_node,int no_ld,double *pt1,struct load *pt2,struct dof *pt3);
int active_col_solver(double *pt_k,int *pt_h,double *pt_r,int n_eq,int check1);
void cal_stress(int order_index,double lf,int no_ele,struct co_or *pt_co,struct ele_con *pt_ele,double *pt_r,struct
dof *pt_dof,struct m_stress *pt,struct m_stress *pt1);
void int_f_vector(int no_dof,int no_ele,double *pt_x,struct co_or *pt_co,struct ele_con *pt_ele,struct dof
*dof_pt,struct m_stress *pt_s);
void residue_vector(int no_dof,double *pt_r,double *pt_x);
double eu_norm(int no_dof,double *pt_x);
```

```
void rev_coor(int n_node,double *pt_v,struct co_or *pt_co,struct dof *pt_dof);
int plastic_h_check(int no_ele,double fy,struct ele_con *pt_ele,struct m_stress *pt_s);
FILE *fp2;
/* Function for Solving */
void solving(char *fn,int mode,double fy,int no_node,int no_ele,int no_ld,struct co_or *pt_co,struct ele_con
*pt_ele,struct load *pt_ld)
{
struct dof *dof_pt;            /* *dof_pt = Pointer for Point DOF Number In Memory */
struct m_stress *pt_s,*pt_sr,*pt_ss;     /* all Pointers in This Line for Point Stress Value In Memory */
register int i,j,k,count1=0,count2=0;
int no_dof,*pt_h,l=0,m,count,order_index,e_p_index,index1,pos_index=1; /*pos_index=1 positive definite metrix
*/
                                                                    /*pos_index=0 non-positive
definite metrix */
double *pt_k,*pt_r,*pt_x,*pt_v,*pt_vs,norm_r,norm,l_factor,lf_interval,lf_step;
/*      *pt_h = Pointer for Point Column Height of Skyline
        *pt_k = Pointer for Point Stiffness Matrix in 1 Dimensional Dynamic Array
        *pt_r = Pointer for Point External Load Vector and Displacement Vector
        norm_r,norm= Eucledian Norm Value */
double diag_check=1e-15;
FILE *fp1;
char fn1[15],fn2[15];
switch(mode){
  case 0:
        order_index=1; /*1st order: order index=1*/
                       /*2nd order: order index=2*/
        e_p_index=1;   /*Elastic Analysis: e_p_index=1*/
                       /*Elastic-Plastic Analysis: e_p_index=2*/
        lf_interval=1.0;
        break;
  case 1:
        order_index=2;
        e_p_index=1;
        lf_interval=1.0;
        break;
```

```
  case 2:
        order_index=1;
        e_p_index=2;
        lf_interval=0.005;
        break;
  default:
        order_index=2;
        e_p_index=2;
        lf_interval=0.005; /* default load factor interval for E-P analysis */
        break;
}

no_dof=get_dof(no_node,no_ele,pt_co,pt_ele);

pt_s=malloc(no_ele*sizeof(struct m_stress));
if(pt_s==NULL)printf("Out of Memory.\n");

pt_ss=malloc(no_ele*sizeof(struct m_stress));
if(pt_ss==NULL)printf("Out of Memory.\n");

pt_sr=malloc(no_ele*sizeof(struct m_stress));
if(pt_sr==NULL)printf("Out of Memory.\n");      /*allocate memory for internal residue load vector*/


dof_pt=malloc(no_dof*sizeof(struct dof));
if(dof_pt==NULL)printf("Out of Memory.\n");
gen_dof_no(no_node,no_dof,pt_co,dof_pt);

pt_h=alloc_h(no_dof+1);  /*allocate memory for input height index of stiffness metrix*/
get_h(no_dof,pt_h,no_ele,pt_ele,dof_pt);  /*calculate column height*/


pt_k=alloc_K(pt_h[no_dof]); /*allocate memory for input stiffness co-ef*/
pt_r=alloc_K(no_dof);  /*allocate memory for input load vector*/
pt_x=alloc_K(no_dof);  /*allocate memory for internal load vector*/
pt_v=alloc_K(no_dof);  /*allocate memory for external displacement vector*/
```

```
pt_vs=alloc_K(no_dof);   /*allocate memory for external displacement vector*/
l_factor=0; /*load factor initialize */


for(j=0;j<no_dof;j++)pt_v[j]=0;
for(j=0;j<no_dof;j++)pt_vs[j]=0;

strcpy(fn1,fn);
strcpy(fn2,fn);
strcat(fn2,".lfd");
strcat(fn1,".str");

/* open output files*/
if((fp1=fopen(fn2,"w"))==NULL)printf("Cannot open .lfd output file.\n");
if((fp2=fopen(fn1,"w"))==NULL)printf("Cannot open .str output file.\n");

/* write displacement to .lfd output file */
fprintf(fp1,"%i\\\n",no_node);

print_disp(l_factor,fp1,no_node,dof_pt,pt_v);

/*for(i=0;i<no_dof;i++)fprintf(fp1,"%f\\",pt_v[i]);
*/
/* write to .str file */
fprintf(fp2,"%i\\\n",no_ele);
count=0;
m=0;
index1=0;
for(i=0;i<no_ele;i++){
  pt_ss[i].lf=0;
  for(j=0;j<6;j++)pt_ss[i].s[j]=0;
}
lf_step=0;
do{
    l_factor+=lf_interval;
    lf_step+=lf_interval;
```

```
for(i=0;i<no_ele;i++){
        pt_s[i].lf=0;
        for(j=0;j<6;j++)pt_s[i].s[j]=0;
}
for(i=0;i<no_ele;i++){
       pt_sr[i].lf=0;
       for(j=0;j<6;j++)pt_sr[i].s[j]=pt_ss[i].s[j];
}

for(j=0;j<no_dof;j++)pt_v[j]=pt_vs[j];
get_r(no_dof,no_ld,pt_r,pt_ld,dof_pt);
for(j=0;j<no_dof;j++)pt_r[j]*=l_factor;   /*generate total load vector*/
for(j=0;j<no_dof;j++)pt_x[j]=pt_r[j];

get_r(no_dof,no_ld,pt_r,pt_ld,dof_pt);
for(j=0;j<no_dof;j++)pt_r[j]*=lf_step;   /*generate step load vector*/
norm_r=eu_norm(no_dof,pt_r);

do{
 if(order_index==2||index1==0){
        for(i=0;i<pt_h[no_dof];i++)pt_k[i]=0;
        get_k(no_ele,pt_h,pt_k,pt_co,pt_ele,dof_pt,pt_sr,order_index); /*assembling stiffness co-ef*/
 }

 chk_k(no_dof,pt_h,pt_k);

 pos_index=active_col_solver(pt_k,pt_h,pt_r,no_dof,index1);
 if(pos_index==1){
        if(order_index==1)index1++;

        for(i=0;i<no_dof;i++)pt_v[i]+=pt_r[i];

        cal_stress(order_index,l_factor,no_ele,pt_co,pt_ele,pt_r,dof_pt,pt_s,pt_sr);
        int_f_vector(no_dof,no_ele,pt_r,pt_co,pt_ele,dof_pt,pt_sr);
        residue_vector(no_dof,pt_x,pt_r);
```

```
            /*  for(i=0;i<no_dof;i++)pt_x[i]=pt_r[i];
            */
                norm=eu_norm(no_dof,pt_r);
        }
        else{
                norm=-1;
        }
            ++count1;
            gotoxy(5,5);
            printf("%i\\",count1 );
    }while(order_index==2&&norm>=0.1*norm_r/100);
    if(pos_index==1){
            if(l_factor>=1)lf_interval=.0005;
            /* write displacement to .lfd output file */
            if(e_p_index==2)l=plastic_h_check(no_ele,fy,pt_ele,pt_sr);
            if(l==1){
               ++m;
               count=m;
            }
            if(l==1||e_p_index==1){
               print_disp(l_factor,fp1,no_node,dof_pt,pt_v);
/*             rev_coor(no_node,pt_v,pt_co,dof_pt);
*/
               lf_step=0;
               index1=0;
               printf("lf=%f\\",l_factor);
               for(i=0;i<no_ele;i++){
                     pt_ss[i].lf=l_factor;
                     for(j=0;j<6;j++)pt_ss[i].s[j]=pt_sr[i].s[j];
               }
               for(j=0;j<no_dof;j++)pt_vs[j]=pt_v[j];
               fprintf(fp2,"%f\\",l_factor);
               fprintf(fp2,"%i\\",m);
               for(j=0;j<no_ele;j++){
                  if(pt_ele[j].hinge_1==1||pt_ele[j].hinge_2==1){
                        fprintf(fp2,"%i\\",j+1);
```

```
                    if(pt_ele[j].hinge_1==1&&pt_ele[j].hinge_2==0)fprintf(fp2,"N\\");
                    if(pt_ele[j].hinge_1==0&&pt_ele[j].hinge_2==1)fprintf(fp2,"F\\");
                    if(pt_ele[j].hinge_1==1&&pt_ele[j].hinge_2==1)fprintf(fp2,"B\\");
                }
            }
            fprintf(fp2,"-1\\\\n");
            for(i=0;i<no_ele;i++){
                for(j=0;j<6;j++)fprintf(fp2,"%f\\",pt_ss[i].s[j]);
                fprintf(fp2,"\n");
            }
        }
    }
++count2;
gotoxy(5,6);
printf("%i\\",count2);
}while(e_p_index==2&&pos_index==1);
fprintf(fp1,"-1\\");
fprintf(fp2,"-1\\");
fclose(fp1);
fclose(fp2);
free(pt_s);
free(pt_ss);
free(dof_pt);
free(pt_h);
free(pt_k);
free(pt_r);
free(pt_x);
free(pt_v);
free(pt_vs);
free(pt_sr);
return;
}

int get_dof(int no_node;int no_ele,struct co_or *pt_co,struct ele_con *pt_ele)
{
int j,i,no_dof,no_free,no_hing;
```

```
no_free=0;
no_hing=0;
for(i=0;i<no_node;i++)no_free=no_free+pt_co[i].bou[0]+pt_co[i].bou[1]+pt_co[i].bou[2];
for(i=0;i<no_ele;i++)no_hing=no_hing+pt_ele[i].hinge_1+pt_ele[i].hinge_2;
no_dof=no_free;
return(no_dof);
}
void gen_dof_no(int no_node,int no_dof,struct co_or *pt_co,struct dof *dof_pt)
{register int i;
 int j=0;
 for(i=0;i<no_node;i++){
    if(pt_co[i].bou[0]==1)dof_pt[i].x_dof=++j;
    else dof_pt[i].x_dof=0;
    if(pt_co[i].bou[1]==1)dof_pt[i].y_dof=++j;
    else dof_pt[i].y_dof=0;
    if(pt_co[i].bou[2]==1)dof_pt[i].theta_dof=++j;
    else dof_pt[i].theta_dof=0;
    if(j>no_dof)printf("data error\n");
 }
return;
}
int *alloc_h(int n)
{
int *pt;
pt=(int *)malloc(n*sizeof(int));
if(pt==NULL){
   printf("Out of Memory.\n");
   exit(1);
}
return(pt);
}
void gen_lm(int ele_no,int *lm,struct ele_con *pt_ele,struct dof *pt_dof);
void cal_h(int *pt_h,int *lm);
void get_h(int no_dof,int *pt_h,int no_ele,struct ele_con *pt_ele,struct dof *pt_dof)  /* Calculate Height of Each
Column */
{
```

```
register i;
int lm[6],*h;
h=(int *)malloc(no_dof*sizeof(int));
if(h==NULL){
   printf("Out of Memory.\n");
   exit(1);
}
  for(i=0;i<no_dof;i++)h[i]=0;
  for(i=0;i<no_ele;i++){
     gen_lm(i,lm,pt_ele,pt_dof);
     cal_h(h,lm);
  }
pt_h[0]=0;
for(i=0;i<no_dof;i++)pt_h[i+1]=pt_h[i]+h[i];
free(h);
return;
}
void gen_lm(int i,int *lm,struct ele_con *pt_ele,struct dof *pt_dof)
{
  lm[0]=pt_dof[pt_ele[i].node_1-1].x_dof;
  lm[1]=pt_dof[pt_ele[i].node_1-1].y_dof;
  lm[2]=pt_dof[pt_ele[i].node_1-1].theta_dof;
  lm[3]=pt_dof[pt_ele[i].node_2-1].x_dof;
  lm[4]=pt_dof[pt_ele[i].node_2-1].y_dof;
  lm[5]=pt_dof[pt_ele[i].node_2-1].theta_dof;
  return;
}
void cal_h(int *pt_h,int *lm)
{
register int i,j,k;
int tempt_h;
  for(i=0;i<6;i++){
     j=lm[i];
     for(k=0;k<6;k++){
          if(k!=j){
             if((j!=0)&&(lm[k]!=0)&&(lm[k]<j)){
```

```
                tempt_h=j-lm[k];
                if(tempt_h>pt_h[j-1]-1)pt_h[j-1]=tempt_h+1;
              }
            }else
            {
                tempt_h=0;
                if(tempt_h>pt_h[j-1]-1)pt_h[j-1]=tempt_h+1;

            }
      }
  }
return;
}
double *alloc_K(int h)
{
double *pt;
pt=(double *)malloc(h*sizeof(double));
if(pt==NULL){
   printf("Out of Memory.\n");
   exit(1);
}
return(pt);
}
void form_stiff(int i,double *st_m,double l,double E,double I,double A,double n);
void rev_stiff(int n,double *st_m);
void rotat(double *st_m,double c,double s);
void assembling(double *st_m,int *lm,int *pt_h,double *pt_k);
void get_k(int no_ele,int *pt_h,double *pt_k,struct co_or *pt_co,struct ele_con *pt_ele,struct dof *pt_dof,struct
m_stress *pt_s,int order)
{
  register int i;
  int lm[6];
  double l,x[2],y[2],st_m[21],n,c,s;
  for(i=0;i<no_ele;i++){
     if(order==2)n=pt_s[i].s[0];  /* Normal Force*/
     else n=0;
```

```
    x[0]=pt_co[pt_ele[i].node_1-1].x;
    x[1]=pt_co[pt_ele[i].node_2-1].x;
    y[0]=pt_co[pt_ele[i].node_1-1].y;
    y[1]=pt_co[pt_ele[i].node_2-1].y;
    l=sqrt((x[0]-x[1])*(x[0]-x[1])+(y[0]-y[1])*(y[0]-y[1]));
    c=(x[1]-x[0])/l;
    s=(y[1]-y[0])/l;
    form_stiff(i,st_m,l,pt_ele[i].Es,pt_ele[i].I,pt_ele[i].A,n);
    if((pt_ele[i].hinge_1==1)&&(pt_ele[i].hinge_2==1)){
        rev_stiff(2,st_m);
        rev_stiff(5,st_m);
    }
    if((pt_ele[i].hinge_1==1)&&(pt_ele[i].hinge_2==0)){
        rev_stiff(2,st_m);
    }
    if((pt_ele[i].hinge_1==0)&&(pt_ele[i].hinge_2==1)){
        rev_stiff(5,st_m);
    }
    rotat(st_m,c,s);
    gen_lm(i,lm,pt_ele,pt_dof);
    assembling(st_m,lm,pt_h,pt_k);
  }
return;
}
void form_stiff(int i,double *st_m,double l,double E,double I,double A,double n)  /* Calculate Local Stiffness */
{   double n_,phi[4];
    n_=-n*l*l/E/I;
    phi[0]=1.+n_/10.-n_*n_/8400.+n_*n_*n_/756000.;
    phi[1]=1.+n_/60.-n_*n_/8400.+n_*n_*n_/756000.;
    phi[2]=1.+n_/30.-n_*n_*11./25200.+n_*n_*n_/108000.;
    phi[3]=1.-n_/60.+n_*n_*13./25200.-n_*n_*n_*11/756000.;
    /* generate member stiffness */
    st_m[0]=E*A/l;
    st_m[1]=12.0*(E*I/l/l/l)*phi[0];
    st_m[2]=0.0;
    st_m[3]=4.0*(E*I/l)*phi[2];
```

```
    st_m[4]=6.0*(E*I/l/l)*phi[1];
    st_m[5]=0.0;
    st_m[6]=E*A/l;
    st_m[7]=0.0;
    st_m[8]=0.0;
    st_m[9]=-E*A/l;
    st_m[10]=12.0*(E*I/l/l/l)*phi[0];
    st_m[11]=0.0;
    st_m[12]=-6.0*(E*I/l/l)*phi[1];
    st_m[13]=-12.0*(E*I/l/l/l)*phi[0];
    st_m[14]=0.0;
    st_m[15]=4.0*(E*I/l)*phi[2];
    st_m[16]=-6.0*(E*I/l/l)*phi[1];
    st_m[17]=0.0;
    st_m[18]=2.0*(E*I/l)*phi[3];
    st_m[19]=6.0*(E*I/l/l)*phi[1];
    st_m[20]=0.0;
return;
}
void rev_stiff(int n,double *st_m)  /* Revise Stiffness As Per End Hinge ( all Real Hinge and Plastic Hinge */
{register int i,j;
int l,m;
 double v[6];
 m=0;
 for(i=0;i<=n;i++)m=m+i;
 l=n;
 for(i=0;i<=n;i++){
    v[i]=st_m[m+l];
    l--;
 }
 j=0;
 for(i=n+1;i<6;i++){
   m=m+i;
   j++;
   v[i]=st_m[m+j];
 }
```

```
 m=0;
 for(i=0;i<6;i++){
   l=i;
   m=m+i;
   for(j=0;j<=i;j++){
     st_m[m+l]=st_m[m+l]-v[j]*v[i]/v[n];
     l--;
   }
 }
 return;
}
void at_k_a(double k[][],double c,double s);
void rotat(double *st_m,double c,double s) /* Rotate Locate Stiffness Projected on Global Stiffness*/
{
double k1[3][3];
register int i,j,k,diag_index;
/* at*k1*a */
diag_index=0;
for(i=0;i<3;i++){
    diag_index=diag_index+i;
    k=0;
    for(j=i;j>=0;j--){
      k1[i][j]=st_m[diag_index+k];
      k=k++;
    }
    for(j=0;j<i;j++)k1[j][i]=k1[i][j];
}
at_k_a(k1,c,s);
diag_index=0;
for(i=0;i<3;i++){
    diag_index=diag_index+i;
    k=0;
    for(j=i;j>=0;j--){
      st_m[diag_index+k]=k1[i][j];
      k++;
    }
```

```
}
/* at*k2*a */
diag_index=3;
for(i=0;i<3;i++){
    diag_index=diag_index+i+3;
    k=diag_index+i+1;
    for(j=2;j>=0;j--){
      k1[j][i]=st_m[k];
      k++;
    }
}
at_k_a(k1,c,s);
diag_index=3;
for(i=0;i<3;i++){
    diag_index=diag_index+i+3;
    k=diag_index+i+1;
    for(j=2;j>=0;j--){
      st_m[k]=k1[j][i];
      k++;
    }
}
/* at*k3*a */
diag_index=3;
for(i=0;i<3;i++){
    diag_index=diag_index+i+3;
    k=0;
    for(j=i;j>=0;j--){
      k1[i][j]=st_m[diag_index+k];
      k=k++;
    }
    for(j=0;j<i;j++)k1[j][i]=k1[i][j];
}
at_k_a(k1,c,s);
diag_index=3;
for(i=0;i<3;i++){
    diag_index=diag_index+i+3;
```

```
    k=0;
    for(j=i;j>=0;j--){
      st_m[diag_index+k]=k1[i][j];
      k++;
    }
}
return;
}
void form_a(double a[][],double c,double s);
void a_transpose(double a[][]);
void mul(double a[][],double b[][]);
void at_k_a(double k1[3][3],double c,double s)
{
register int i,j;
double a[3][3];
form_a(a,c,s);
mul(k1,a);
a_transpose(a);
mul(a,k1);
  for(i=0;i<3;i++){
    for(j=0;j<3;j++)k1[i][j]=a[i][j];
  }
return;
}
void form_a(double a[3][3],double c,double s)
{
a[0][0]=c;
a[0][1]=s;
a[0][2]=0.;
a[1][0]=-s;
a[1][1]=c;
a[1][2]=0.;
a[2][0]=0.;
a[2][1]=0.;
a[2][2]=1.;
return;
```

```
}
void a_transpose(double a[3][3])
{
register int i,j;
double aa[3][3];
  for(i=0;i<3;i++){
    for(j=0;j<3;j++)aa[i][j]=a[j][i];
  }
  for(i=0;i<3;i++){
    for(j=0;j<3;j++)a[i][j]=aa[i][j];
  }
return;
}
void mul(double a[3][3],double b[3][3])
{
register int i,j,k;
double ab[3][3],sum;
  for(i=0;i<3;i++){
     for(j=0;j<3;j++){
           sum=0;
           for(k=0;k<3;k++)sum=sum+a[i][k]*b[k][j];
           ab[i][j]=sum;
     }
  }
  for(i=0;i<3;i++){
    for(j=0;j<3;j++)a[i][j]=ab[i][j];
  }
return;
}
void assembling(double *st_m,int *lm,int *pt_h,double *pt_k)  /* Assembling Stiffness */
{
register int i,j,k,l,m;
m=0;
for(i=0;i<6;i++){
   k=0;
   m=m+i;
```

```
    if(lm[i]!=0){
      for(j=i;j>=0;j--){
          if(lm[j]!=0){
            l=lm[i]-lm[j];
            pt_k[pt_h[lm[i]-1]+l]=pt_k[pt_h[lm[i]-1]+l]+st_m[m+k];
            k++;
          }
          continue;
      }
    }
    continue;
}
return;
}
void chk_k(int no_dof,int *pt_h,double *pt_k)
{
register int i;
int j=0;
for(i=0;i<no_dof;i++){
   if(pt_k[pt_h[i]]==0)pt_k[pt_h[i]]=1;
}
return;
}
void get_r(int n_dof,int no_ld,double *pt_r,struct load *pt_ld,struct dof *pt_dof)
{
  int i,k,l,m,n;
  for(i=0;i<n_dof;i++)pt_r[i]=0.;
  for(i=0;i<no_ld;i++){
     k=pt_ld[i].node_no;
     l=pt_dof[k-1].x_dof;
     m=pt_dof[k-1].y_dof;
     n=pt_dof[k-1].theta_dof;
     pt_r[l-1]=pt_ld[i].x_load;
     pt_r[m-1]=pt_ld[i].y_load;
     pt_r[n-1]=pt_ld[i].moment;
  }
```

```
return;
}
int decompose(int n_eq,int *pt_h,double *pt_k);
void forward_reduce(int n_eq,int *pt_h,double *pt_k,double *pt_r);
void back_substi(int n_eq,int *pt_h,double *pt_k,double *pt_r);
/* Function for Simultaneous Linear Equation by LDLt Algoritm */
int active_col_solver(double *pt_k,int *pt_h,double *pt_r,int n_eq,int index1)
{
int h=1;

  if(index1==0)h=decompose(n_eq,pt_h,pt_k);
  if(h==1){
     forward_reduce(n_eq,pt_h,pt_k,pt_r);
     back_substi(n_eq,pt_h,pt_k,pt_r);
  }
return(h);
}
int decompose(int n_eq,int *pt_h,double *pt_k)
{
register int h,i,j,k,p,q,r,i_col,h_i,u,check=1;
double b,c;
for(i=0;i<n_eq;i++){
  h=pt_h[i+1]-pt_h[i]-2;
  if(h>0){
     p=i-h;
     r=pt_h[i+1]-1;
     for(j=0;j<h;j++){
       r--;
       i_col=pt_h[p];
       h_i=pt_h[p+1]-pt_h[p]-2;
       u=min(j,h_i);
       c=0.;
       for(k=0;k<=u;k++){
            c=c+pt_k[i_col+k+1]*pt_k[r+k+1];
       }
       pt_k[r]=pt_k[r]-c;
```

```
      p++;
    }
  }
    p=i;
    b=0.;
    for(j=pt_h[i]+1;j<=pt_h[i+1]-1;j++){
      p--;
      i_col=pt_h[p];
      c=pt_k[j]/pt_k[i_col];
      b=b+c*pt_k[j];
      pt_k[j]=c;
    }
    pt_k[pt_h[i]]=pt_k[pt_h[i]]-b;
    if(pt_k[pt_h[i]]<=0){
/*        printf("stiffness matrix is negative definite\n");
          printf("%f\\",pt_k[pt_h[i]]);
          */
          if(pt_k[pt_h[i]]==0)pt_k[pt_h[i]]=-1;
          check++;
    }
if(check!=1)i=n_eq;
}
 if(check==1)return(1);
 else return(0);
}
void forward_reduce(int n_eq,int *pt_h,double *pt_k,double *pt_r)
{
register int i,j,k;
double c;
for(i=0;i<n_eq;i++){
   if(pt_h[i+1]-pt_h[i]-2>=0){
     j=i;
     c=0.;
     for(k=pt_h[i]+1;k<=pt_h[i+1]-1;k++){
       j--;
       c=c+pt_k[k]*pt_r[j];
```

```
    }
    pt_r[i]=pt_r[i]-c;
  }
  continue;
}
return;
}
void back_substi(int n_eq,int *pt_h,double *pt_k,double *pt_r)
{
register int i,j,k,l,u,p,q;
for(i=0;i<n_eq;i++)pt_r[i]=pt_r[i]/pt_k[pt_h[i]];
p=n_eq-1;
for(i=1;i<n_eq;i++){
  l=pt_h[p]+1;
  u=pt_h[p+1]-1;
  q=p;
  for(j=l;j<=u;j++){
    q--;
    pt_r[q]=pt_r[q]-pt_k[j]*pt_r[p];
  }
  p--;
}
return;
}
cal_local_disp(int *lm,double *pt_r,double *v,double c,double s);
void stress(double *st_m,double *v);
/* Calculate Internal Member Force */
void cal_stress(int order,double l_f,int no_ele,struct co_or *pt_co,struct ele_con *pt_ele,double *pt_r,struct dof
*pt_dof,struct m_stress *pt_s,struct m_stress *pt_sr)
{
  register int i,j,k;
  int lm[6],m,an;
  double l,x[2],y[2],st_m[21],n,c,s,v[6];
  for(i=0;i<no_ele;i++){
    if(order==2)n=pt_sr[i].s[0];  /* Normal Force*/
    else n=0;
```

```
    x[0]=pt_co[pt_ele[i].node_1-1].x;
    x[1]=pt_co[pt_ele[i].node_2-1].x;
    y[0]=pt_co[pt_ele[i].node_1-1].y;
    y[1]=pt_co[pt_ele[i].node_2-1].y;
    l=sqrt((x[0]-x[1])*(x[0]-x[1])+(y[0]-y[1])*(y[0]-y[1]));
    c=(x[1]-x[0])/l;
    s=(y[1]-y[0])/l;
    an=0;
    do{
      for(k=0;k<6;k++)v[k]=0;
      gen_lm(i,lm,pt_ele,pt_dof);
      cal_local_disp(lm,pt_r,v,c,s);  /* local displacement at each load step */
      form_stiff(i,st_m,l,pt_ele[i].Es,pt_ele[i].I,pt_ele[i].A,n);
      if((pt_ele[i].hinge_1==1)&&(pt_ele[i].hinge_2==1)){
            rev_stiff(2,st_m);
            rev_stiff(5,st_m);
      }
      if((pt_ele[i].hinge_1==1)&&(pt_ele[i].hinge_2==0))rev_stiff(2,st_m);
      if((pt_ele[i].hinge_1==0)&&(pt_ele[i].hinge_2==1)){
           rev_stiff(5,st_m);
      }
      stress(st_m,v);
/*          for(m=0;m<6;m++)fprintf(fp2,"ELE=%i,intf[%i]=%f\\",i+1,m+1,v[m]);
            fprintf(fp2,"\n");
*/          n+=v[0];
            ++an;
    }while(order==2&&an<2);
    for(j=0;j<6;j++)pt_sr[i].s[j]+=v[j];
/*    for(j=0;j<6;j++)printf("intf[%i][%i]=%f\\",i,j,pt_sr[i].s[j]);
    getch();
  */
    pt_sr[i].lf=l_f;
    for(j=0;j<6;j++)pt_s[i].s[j]=v[j];
  }
return;
}
```

```
void mul_a_v(double a[3][3],double *v);
cal_local_disp(int *lm,double *pt_r,double *v,double c,double s)
{register int i,j;
 double l,a[3][3];
 for(i=0;i<6;i++){
   if(lm[i]!=0)v[i]=pt_r[lm[i]-1];
   else v[i]=0;
 }
 form_a(a,c,s);
 mul_a_v(a,&v[0]);
 mul_a_v(a,&v[3]);
 return;
}
void mul_a_v(double a[3][3],double *v)
{
 register int i,j;
 double s[3],l;
 for(j=0;j<3;j++)s[j]=v[j];
 for(i=0;i<3;i++){
    l=0;
    for(j=0;j<3;j++)l+=a[i][j]*s[j];
    v[i]=l;
 }
return;
}
void stress(double *st_m,double *v)
{
register int i,j,k,m,n;
double l,s[6];
m=0;
for(i=0;i<6;i++){
  l=0;
  k=i;
  m=m+i;
  for(j=0;j<i;j++){
    l=l+st_m[m+k]*v[j];
```

```
    k--;
  }
  n=m;
  k=0;
  for(j=i;j<6;j++){
    if(j>i)m+=j;
    l=l+st_m[m+k]*v[j];
    k++;
  }
  s[i]=l;
  m=n;
}
for(i=0;i<6;i++)v[i]=s[i];
return;
}
void int_f_vector(int no_dof,int no_ele,double *pt_x,struct co_or *pt_co,struct ele_con *pt_ele,struct dof
*pt_dof,struct m_stress *pt_s)
{
  register int i,j,k;
  int lm[6];
  double l,x[2],y[2],st_m[21],a[3][3],c,s,v[6];
  for(i=0;i<no_dof;i++)pt_x[i]=0;
  for(i=0;i<no_ele;i++){
     x[0]=pt_co[pt_ele[i].node_1-1].x;
     x[1]=pt_co[pt_ele[i].node_2-1].x;
     y[0]=pt_co[pt_ele[i].node_1-1].y;
     y[1]=pt_co[pt_ele[i].node_2-1].y;
     l=sqrt((x[0]-x[1])*(x[0]-x[1])+(y[0]-y[1])*(y[0]-y[1]));
     c=(x[1]-x[0])/l;
     s=(y[1]-y[0])/l;
     form_a(a,c,s);
     a_transpose(a);
     for(j=0;j<6;j++)v[j]=pt_s[i].s[j];
     mul_a_v(a,&v[0]);
     mul_a_v(a,&v[3]);
     gen_lm(i,lm,pt_ele,pt_dof);
```

```
    for(j=0;j<6;j++)if(lm[j]!=0)pt_x[lm[j]-1]+=v[j];
  }
return;
}
void residue_vector(int no_dof,double *pt_r,double *pt_x)
{
register int i;
for(i=0;i<no_dof;i++)pt_x[i]=pt_r[i]-pt_x[i];
return;
}
double eu_norm(int n,double *pt)
{
double a;
register int i;
a=0;
for(i=0;i<n;i++)a+=pt[i]*pt[i];
return(sqrt(a));
}

int plastic_m_index(double fy,double A,double Z,double axial_f,double end_m);
void updat_h1(int i,struct ele_con *pt);
void updat_h2(int i,struct ele_con *pt);
/* Check Plastic Hinge Formation  */
int plastic_h_check(int no_ele,double fy,struct ele_con *pt_ele,struct m_stress *pt_s)
{
register int i,j,k,l=0;
double axial_f,end_m1,end_m2;
  for(i=0;i<no_ele;i++){
    axial_f=pt_s[i].s[0];
    end_m1=pt_s[i].s[2];
    end_m2=pt_s[i].s[5];
    j=plastic_m_index(fy,pt_ele[i].A,pt_ele[i].Z,axial_f,end_m1);
    k=plastic_m_index(fy,pt_ele[i].A,pt_ele[i].Z,axial_f,end_m2);
    if(j!=0&&j!=1)printf("data error\n");
    if(k!=0&&k!=1)printf("data error\n");
    if(pt_ele[i].hinge_1==0){
```

```
      if(j==1){
           updat_h1(j,&pt_ele[i]);
           l=1;
      }
    }
    if(pt_ele[i].hinge_2==0){
      if(k==1){
           updat_h2(k,&pt_ele[i]);
           l=1;
      }
    }
  }
return(l);
}
/* Check Plastic Hinge Formation with AISC Interaction Equation */
int plastic_m_index(double fy,double A,double Z,double axial_f,double end_m)
{
register int i;
double p_yield,plastic_m;
  p_yield=fy*A;
  plastic_m=fy*Z;
  if(fabs(axial_f)/p_yield<=.15){
    if(fabs(end_m)<plastic_m)i=0;
    else i=1;
  }
  else{
    if(fabs(end_m)<1.18*(1-fabs(axial_f)/p_yield)*plastic_m)i=0;
    else i=1;
  }
return(i);
}
void updat_h1(int i,struct ele_con *pt)
{
  pt->hinge_1=i;
}
void updat_h2(int i,struct ele_con *pt)
```

```
{
  pt->hinge_2=i;
}
void rev_coor(int n_node,double *pt_v,struct co_or *pt_co,struct dof *pt_dof)
{
int i;
  for(i=0;i<n_node;i++){
    if(pt_dof[i].x_dof!=0)pt_co[i].x+=pt_v[pt_dof[i].x_dof];
    if(pt_dof[i].y_dof!=0)pt_co[i].y+=pt_v[pt_dof[i].y_dof];
  }
return;
}
void print_disp(double lf,FILE *fp,int no_node,struct dof *dof_pt,double *pt_v)
{
register int i;
fprintf(fp,"%f\\",lf);
for(i=0;i<no_node;i++){
  if(dof_pt[i].x_dof!=0)fprintf(fp,"%f\\",pt_v[dof_pt[i].x_dof-1]);
  else fprintf(fp,"0\\");
  if(dof_pt[i].y_dof!=0)fprintf(fp,"%f\\",pt_v[dof_pt[i].y_dof-1]);
  else fprintf(fp,"0\\");
  if(dof_pt[i].theta_dof!=0)fprintf(fp,"%f\\",pt_v[dof_pt[i].theta_dof-1]);
  else fprintf(fp,"0\\");
}
fprintf(fp,"\n");
}
```

ภาคผนวก ค

ตัวอย่าง Input และ Output

Project Name : exam3
Coordinate Input Data

lo of Node = 14

le No:	X-Coor	Y-Coor	X-Bou	Y-Bou	Z-Bou
1	0.000	0.000	Fix	Fix	Fix
2	240.000	0.000	Fix	Fix	Fix
3	0.000	300.000			
4	120.000	300.000			
5	240.000	300.000			
6	0.000	600.000			
7	120.000	600.000			
8	240.000	600.000			
9	0.000	900.000			
10	120.000	900.000			
11	240.000	900.000			
12	0.000	1200.000			
13	120.000	1200.000			
14	240.000	1200.000			

Project Name : exam3
Element Connectivity Input Data

No of Element = 16
Yield Strength = 36.000
Modulus of Elasticity = 30000.000

No	Node1	Node2	I	Zp	Area	End1 Con	End2 Con
1	1	3	144.000	53.330	11.750	-	-
2	3	6	144.000	53.330	11.750	-	-
3	6	9	144.000	53.330	11.750	-	-
4	9	12	144.000	53.330	11.750	-	-
5	2	5	144.000	53.330	11.750	-	-
6	5	8	144.000	53.330	11.750	-	-
7	8	11	144.000	53.330	11.750	-	-
8	11	14	144.000	53.330	11.750	-	-
9	3	4	144.000	53.330	11.750	-	-
10	4	5	144.000	53.330	11.750	-	-
11	6	7	144.000	53.330	11.750	-	-
12	7	8	144.000	53.330	11.750	-	-
13	9	10	144.000	53.330	11.750	-	-
14	10	11	144.000	53.330	11.750	-	-
15	12	13	144.000	53.330	11.750	-	-
16	13	14	144.000	53.330	11.750	-	-

:- Project Name : exam3
:- External Load Data

Node No:	X-Load :	Y-Load :	Moment :
3	2.000	-10.000	0.000
4	0.000	-20.000	0.000
5	0.000	-10.000	0.000
6	2.000	-10.000	0.000
7	0.000	-20.000	0.000
8	0.000	-10.000	0.000
9	2.000	-10.000	0.000
10	0.000	-20.000	0.000
11	0.000	-10.000	0.000
12	2.000	-10.000	0.000
13	0.000	-20.000	0.000
14	0.000	-10.000	0.000

:- Project Name : exam3
:- Internal Member Force

Load Factor= 1.108 :Hinge No=1
Ele No=10:End=Far\

EleNo:	Axial	Shear	Bending	Axial	Shear	Bending
1	6.074e+01	3.779e+00	9.542e+02	-6.074e+01	-3.779e+00	5.199e+02
2	5.023e+01	1.598e+00	3.550e+02	-5.023e+01	-1.598e+00	4.430e+02
3	3.720e+01	6.425e-01	1.044e+02	-3.720e+01	-6.425e-01	2.396e+02
4	2.019e+01	-1.444e+00	-1.930e+02	-2.019e+01	1.444e+00	-2.003e+02
5	1.165e+02	5.085e+00	1.157e+03	-1.165e+02	-5.085e+00	9.463e+02
6	8.273e+01	5.050e+00	9.745e+02	-8.273e+01	-5.050e+00	1.024e+03
7	5.144e+01	3.790e+00	6.198e+02	-5.144e+01	-3.790e+00	7.173e+02
8	2.413e+01	3.660e+00	4.717e+02	-2.413e+01	-3.660e+00	6.731e+02
9	3.528e-02	-5.689e-01	-8.749e+02	-3.528e-02	5.689e-01	8.068e+02
10	3.528e-02	-2.273e+01	-8.068e+02	-3.528e-02	2.273e+01	-1.921e+03
11	1.260e+00	1.951e+00	-5.474e+02	-1.260e+00	-1.951e+00	7.824e+02
12	1.260e+00	-2.021e+01	-7.824e+02	-1.260e+00	2.021e+01	-1.643e+03
13	1.292e-01	5.931e+00	-4.658e+01	-1.292e-01	-5.931e+00	7.585e+02
14	1.292e-01	-1.623e+01	-7.585e+02	-1.292e-01	1.623e+01	-1.189e+03
15	3.660e+00	9.109e+00	2.003e+02	-3.660e+00	-9.109e+00	8.957e+02
16	3.660e+00	-1.305e+01	-8.957e+02	-3.660e+00	1.305e+01	-6.731e+02

Load Factor= 1.229 :Hinge No=2
Ele No=10:End=Far\Ele No=12:End=Far\

EleNo:	Axial	Shear	Bending	Axial	Shear	Bending
1	6.650e+01	4.847e+00	1.256e+03	-6.650e+01	-4.847e+00	6.559e+02
2	5.460e+01	2.269e+00	4.628e+02	-5.460e+01	-2.269e+00	6.443e+02
3	4.109e+01	6.295e-01	8.987e+01	-4.109e+01	-6.295e-01	2.765e+02
4	2.236e+01	-1.584e+00	-2.112e+02	-2.236e+01	1.584e+00	-2.187e+02
5	1.301e+02	4.981e+00	1.359e+03	-1.301e+02	-4.981e+00	9.421e+02
6	9.282e+01	5.102e+00	9.787e+02	-9.282e+01	-5.102e+00	1.218e+03
7	5.719e+01	4.284e+00	7.019e+02	-5.719e+01	-4.284e+00	8.201e+02
8	2.678e+01	4.041e+00	5.180e+02	-2.678e+01	-4.041e+00	7.478e+02
9	-1.216e-01	-3.798e-01	-1.119e+03	1.216e-01	3.798e-01	1.073e+03
10	-1.216e-01	-2.495e+01	-1.073e+03	1.216e-01	2.495e+01	-1.921e+03
11	8.177e-01	1.225e+00	-7.342e+02	-8.177e-01	-1.225e+00	8.822e+02
12	8.177e-01	-2.334e+01	-8.822e+02	-8.177e-01	2.334e+01	-1.920e+03
13	2.430e-01	6.437e+00	-6.531e+01	-2.430e-01	-6.437e+00	8.379e+02
14	2.430e-01	-1.813e+01	-8.379e+02	-2.430e-01	1.813e+01	-1.338e+03
15	4.041e+00	1.008e+01	2.187e+02	-4.041e+00	-1.008e+01	9.941e+02
16	4.041e+00	-1.449e+01	-9.941e+02	-4.041e+00	1.449e+01	-7.478e+02

Load Factor= 1.291 :Hinge No=3
Ele No=5:End=Near\Ele No=10:End=Far\Ele No=12:End=Far\

EleNo:	Axial	Shear	Bending	Axial	Shear	Bending
1	6.800e+01	5.618e+00	1.509e+03	-6.800e+01	-5.618e+00	7.429e+02
2	5.606e+01	3.237e+00	6.715e+02	-5.606e+01	-3.237e+00	8.736e+02
3	4.234e+01	9.514e-01	1.122e+02	-4.234e+01	-9.514e-01	4.092e+02
4	2.341e+01	-1.713e+00	-2.416e+02	-2.341e+01	1.713e+00	-2.206e+02
5	1.386e+02	4.714e+00	1.524e+03	-1.386e+02	-4.714e+00	9.162e+02
6	9.892e+01	4.512e+00	1.005e+03	-9.892e+01	-4.512e+00	1.274e+03
7	6.098e+01	4.215e+00	6.461e+02	-6.098e+01	-4.215e+00	9.388e+02
8	2.825e+01	4.296e+00	5.488e+02	-2.825e+01	-4.296e+00	8.010e+02
9	2.018e-01	-9.816e-01	-1.414e+03	-2.018e-01	9.816e-01	1.297e+03
10	2.018e-01	-2.681e+01	-1.297e+03	-2.018e-01	2.681e+01	-1.921e+03
11	2.974e-01	8.069e-01	-9.858e+02	-2.974e-01	-8.069e-01	1.084e+03
12	2.974e-01	-2.502e+01	-1.084e+03	-2.974e-01	2.502e+01	-1.920e+03

13	-8.160e-02	6.018e+00	-1.676e+02	8.160e-02	-6.018e+00	8.899e+02
14	-8.160e-02	-1.981e+01	-8.899e+02	8.160e-02	1.981e+01	-1.488e+03
15	4.296e+00	1.049e+01	2.206e+02	-4.296e+00	-1.049e+01	1.042e+03
16	4.296e+00	-1.534e+01	-1.042e+03	-4.296e+00	1.534e+01	-8.010e+02

Load Factor= 1.332 :Hinge No=4

Ele No=1:End=Near\Ele No=5:End=Near\Ele No=10:End=Far\Ele No=12:End=Far\

EleNo:	Axial	Shear	Bending	Axial	Shear	Bending
1	6.761e+01	7.140e+00	1.907e+03	-6.761e+01	-7.140e+00	9.511e+02
2	5.639e+01	4.004e+00	8.297e+02	-5.639e+01	-4.004e+00	1.091e+03
3	4.287e+01	1.274e+00	1.374e+02	-4.287e+01	-1.274e+00	5.350e+02
4	2.405e+01	-1.815e+00	-2.683e+02	-2.405e+01	1.815e+00	-2.188e+02
5	1.455e+02	3.516e+00	1.524e+03	-1.455e+02	-3.516e+00	8.764e+02
6	1.034e+02	3.988e+00	1.044e+03	-1.034e+02	-3.988e+00	1.342e+03
7	6.369e+01	4.054e+00	5.779e+02	-6.369e+01	-4.054e+00	1.039e+03
8	2.923e+01	4.479e+00	5.709e+02	-2.923e+01	-4.479e+00	8.404e+02
9	-4.715e-01	-2.104e+00	-1.781e+03	4.715e-01	2.104e+00	1.528e+03
10	-4.715e-01	-2.874e+01	-1.528e+03	4.715e-01	2.874e+01	-1.921e+03
11	-6.616e-02	2.016e-01	-1.228e+03	6.616e-02	-2.016e-01	1.254e+03
12	-6.616e-02	-2.644e+01	-1.254e+03	6.616e-02	2.644e+01	-1.920e+03
13	-4.251e-01	5.501e+00	-2.667e+02	4.251e-01	-5.501e+00	9.269e+02
14	-4.251e-01	-2.114e+01	-9.269e+02	4.251e-01	2.114e+01	-1.610e+03
15	4.479e+00	1.073e+01	2.188e+02	-4.479e+00	-1.073e+01	1.072e+03
16	4.479e+00	-1.591e+01	-1.072e+03	-4.479e+00	1.591e+01	-8.404e+02

:- Project Name : exam3
:- Nodal Displacement

Load Factor= 0.000

Node No:	X-Disp	Y-Disp	Rotation
1	0.000000e+00	0.000000e+00	0.000000e+00
2	0.000000e+00	0.000000e+00	0.000000e+00
3	0.000000e+00	0.000000e+00	0.000000e+00
4	0.000000e+00	0.000000e+00	0.000000e+00
5	0.000000e+00	0.000000e+00	0.000000e+00
6	0.000000e+00	0.000000e+00	0.000000e+00
7	0.000000e+00	0.000000e+00	0.000000e+00
8	0.000000e+00	0.000000e+00	0.000000e+00
9	0.000000e+00	0.000000e+00	0.000000e+00
10	0.000000e+00	0.000000e+00	0.000000e+00
11	0.000000e+00	0.000000e+00	0.000000e+00
12	0.000000e+00	0.000000e+00	0.000000e+00
13	0.000000e+00	0.000000e+00	0.000000e+00
14	0.000000e+00	0.000000e+00	0.000000e+00

Load Factor= 1.108

Node No:	X-Disp	Y-Disp	Rotation
1	0.000000e+00	0.000000e+00	0.000000e+00
2	0.000000e+00	0.000000e+00	0.000000e+00
3	5.181692e+00	-5.169600e-02	-1.708300e-02
4	5.181680e+00	-6.813280e-01	6.274000e-03
5	5.181668e+00	-9.918100e-02	-9.199000e-03
6	1.121242e+01	-9.444500e-02	-1.371500e-02
7	1.121200e+01	-6.969530e-01	4.761000e-03
8	1.121157e+01	-1.695890e-01	-7.203000e-03
9	1.517771e+01	-1.261050e-01	-8.685000e-03
10	1.517767e+01	-6.951470e-01	2.497000e-03
11	1.517762e+01	-2.133670e-01	-3.484000e-03
12	1.713500e+01	-1.432870e-01	-8.949000e-03
13	1.713376e+01	-9.416170e-01	7.180000e-04
14	1.713251e+01	-2.339040e-01	3.813000e-03

Load Factor= 1.229

Node No:	X-Disp	Y-Disp	Rotation
1	0.000000e+00	0.000000e+00	0.000000e+00
2	0.000000e+00	0.000000e+00	0.000000e+00
3	6.943182e+00	-5.659700e-02	-2.363400e-02
4	6.943224e+00	-1.053335e+00	6.811000e-03
5	6.943265e+00	-1.106880e-01	-1.855000e-02
6	1.494723e+01	-1.030630e-01	-1.667100e-02
7	1.494695e+01	-7.972860e-01	5.786000e-03
8	1.494667e+01	-1.896860e-01	-8.634000e-03
9	1.954977e+01	-1.380300e-01	-9.715000e-03
10	1.954969e+01	-7.657150e-01	2.831000e-03
11	1.954960e+01	-2.383610e-01	-4.116000e-03
12	2.175338e+01	-1.570630e-01	-9.987000e-03
13	2.175200e+01	-1.045668e+00	7.930000e-04
14	2.175063e+01	-2.611500e-01	4.217000e-03

Load Factor= 1.291

Node No:	X-Disp	Y-Disp	Rotation
1	0.000000e+00	0.000000e+00	0.000000e+00
2	0.000000e+00	0.000000e+00	0.000000e+00
3	8.530028e+00	-5.787100e-02	-3.018500e-02
4	8.529959e+00	-1.388059e+00	7.470000e-03

5	8.529891e+00	-1.179930e-01	-2.743100e-02
6	1.915538e+01	-1.055850e-01	-2.242500e-02
7	1.915528e+01	-1.098640e+00	6.327000e-03
8	1.915517e+01	-2.021770e-01	-1.625100e-02
9	2.513575e+01	-1.416210e-01	-1.132600e-02
10	2.513578e+01	-8.199980e-01	3.363000e-03
11	2.513581e+01	-2.540720e-01	-4.939000e-03
12	2.761106e+01	-1.615440e-01	-1.056400e-02
13	2.760960e+01	-1.094667e+00	8.600000e-04
14	2.760814e+01	-2.781150e-01	4.214000e-03

Load Factor= 1.332

Node No:	X-Disp	Y-Disp	Rotation
1	0.000000e+00	0.000000e+00	0.000000e+00
2	0.000000e+00	0.000000e+00	0.000000e+00
3	1.073050e+01	-5.753800e-02	-3.766900e-02
4	1.073066e+01	-1.750021e+00	8.293000e-03
5	1.073082e+01	-1.238410e-01	-3.595000e-02
6	2.392458e+01	-1.055300e-01	-2.763100e-02
7	2.392460e+01	-1.359525e+00	6.848000e-03
8	2.392463e+01	-2.118830e-01	-2.355900e-02
9	3.116588e+01	-1.420140e-01	-1.275200e-02
10	3.116602e+01	-8.608830e-01	3.827000e-03
11	3.116617e+01	-2.660890e-01	-5.658000e-03
12	3.387040e+01	-1.624800e-01	-1.095800e-02
13	3.386888e+01	-1.124305e+00	9.060000e-04
14	3.386735e+01	-2.909670e-01	4.127000e-03

ประวัติผู้เขียน

นาย วินัย แก้วกุลทล เกิดเมื่อวันที่ 27 สิงหาคม 2509 ที่จังหวัดแพร่ สำเร็จการศึกษาวิศวกรรมศาสตรบัณฑิตจากภาควิชาวิศวกรรมโยธา คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเชียงใหม่ เมื่อปีการศึกษา 2532 เข้าศึกษาหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต ภาควิชาวิศวกรรมโยธา คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา 2534